Xiao Kuang
Logan Miller

Phone Store Database

Project Link: http://web.engr.oregonstate.edu/~millelog/

Draft Review Feedback:

- Review 1:
    - Below are some notes/suggestions, please reach out if you want to discuss or need any clarification.
    - **Website:**
    - I think similar to me, the website is currently in a state showing what options will be available to end users. This is still a draft, but I do notice there are a number of options not available yet, such as every table should have a select option as well as an add option. For example I think you should be able to add a new phone.
    - Once the site is more built out please let me know if you want me to take another look.
    - **Table Creation / Data Insertion:**
    - It is likely a stylistic choice but to ensure a table has a primary key I am a fan of adding the primary key constraint in the create table rather than the alter table. I think alter table is a fine when adding foreign keys. One item I don't know with MariaDB if you need to explicitly note the Key is foreign. In your Alter statements you add a key but do not specify foreign key. Again I think using the Alter for foreign keys makes sense, this way you do not have to worry about table creation order.
    - Another stylistic preference I have is to include the AUTO_INCREMENT when creating the table. This way you do not have to manually number any data you are inserting, which can cut down on human error. The line for the Cust_ID in the create customer table could be:
    - `Cust_ID` int(11) NOT NULL AUTO_INCREMENT,

- ○ **Data Manipulation:**
- ○ Depending you how you implement the website and what options you give to the user you may want to update some of your queries, specifically the SELECT queries. Currently you always return all rows and columns, and that can be a lot more data than always needed.
- ○ Best of luck finishing!

- Review 2:
  - Hello! A few notes on your draft:
  - DMQ
  - - Your "select stuff" section seems a bit scarce. Currently, the only way to lookup things is to get every single item in a table. Part of the specifications for this class is to be able to filter the results of tables, so you will have to add some Selects that allow for looking up orders, or customers, or something by some attribute.
  - DDQ
  - - For "Customer" you set the customer address to be NULL by default, but then you don't allow the zipcode to be null. Your outline doesn't say anything about making the zipcode or address not be null.
  - - For "Employees" you have set the employee address and zip to be NOT NULL, but your outline doesn't say anything about this constraint like it does for the other attributes that must not be null.
  - WEBSITE
  - - The website pages aren't there, and there will need to be some way to search by text or filter results somehow (like with buttons).

a) Project Outline and Database Outline, ERD and Schema Updated Version

- **Outline:**
  - This database will be used to represent the current state of a company that owns a couple of phone stores. There will be information on the customers that come in, the employees, and the products that are out for sale. For this project, assume the company only sells phones and cellular service plans. The practical application of this idea makes it a good candidate for modeling a database after.

Xiao Kuang
Logan Miller

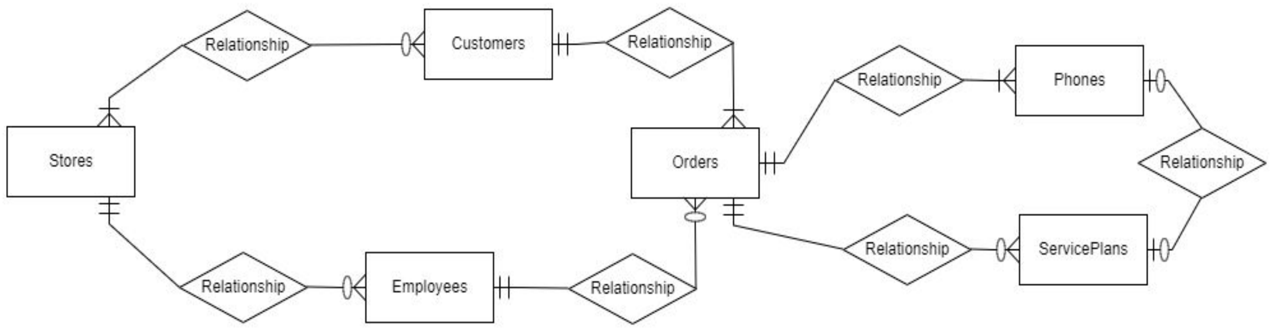
- **Database Outline, in Words**
    - The company needs to keep track of their customers, their records will contain:
        - **Cust_ID**: Primary key, auto-incrementing integer. This will be way to identify each customer uniquely.
        - **Cust_Name**: Customer's name, stored in varchar(255). No need to split into given name and family name for our purposes. Not allowed to be null. Does not have to be unique.
        - **Cust_Phone_Number**: Customer's contact phone number. Does not need to be unique, not allowed to be null. Stored in integer of size 10.
        - **Cust_Email**: Customer's contact email address. Does not need to be unique, not allowed to be null. Stored in varchar(254) (IETF max length for emails[1]). Not allowed to be null. Does not have to be unique.
        - **Cust_Address_Street**: Customer's mailing address, street-line stored in varchar(255). Allowed to be null. Not required.
        - **Cust_Address_Zip**: Customer's zip code associated with the street address. Not allowed to be null. Not unique. Required.
    - Employees make commission off of selling service plans and phones to customers and are tracked as well:
        - **Emp_ID**: Primary key, auto-incrementing integer . This will be way to identify each employee uniquely.
        - **Emp_Name**: Employee's name, stored in varchar(255). No need to split into given name and family name for our purposes. Not allowed to be null. Does not have to be unique.
        - **Store_ID**: Foreign key, associates what store the employee works at. If employee is. There are special store codes associated with corporate and alumni (to signify past-employment).
        - **Emp_Phone_Number**: Employee's contact phone number. Does not need to be unique, not allowed to be null. Stored in integer of size 10.
        - **Emp_Address_Street**: Employee's mailing address, street-line stored in varchar(255). Not allowed to be null. Does not have to be unique.
        - **Emp_Address_Zip**: Employee's zip code associated with the street address. Not allowed to be null. Does not have to be unique.
    - Stores employ employees who sell phones and plans to customers. Each store has:

[1] "What Is the Maximum Length of a Valid Email Address?" *Stack Overflow*, 2008, stackoverflow.com/questions/386294/what-is-the-maximum-length-of-a-valid-email-address.

    - **Store_ID**: Primary key, auto-incrementing. This will identify each store uniquely. Corporate locations and past employees will have special store IDs.
    - **Store_Phone_Number**: Store's contact phone number. Does not need to be unique, not allowed to be null.
    - **Store_Address_Street**: Store's physical address, street-line stored in varchar(255). Not allowed to be null.
    - **Store_Address_Zip**: Store's zip code associated with the street address. Not allowed to be null.
  - Orders are placed when a customer purchases from an employee:
    - **Order_ID**: Primary key, auto-incrementing integer. This will identify each order uniquely.
    - **Cust_ID**: Foreign key, this is the customer that placed the order.
    - **Emp_ID**: Foreign key, this is the employee that sold the order.
    - **Order_Date**: The date and time the order was placed, stored in datetime. Automatically generated.
  - Phones are one of the two things sold in the stores:
    - **IMEI**: Primary key. Serial number on box of phone, uniquely identifies each phone.
    - **SKU**: Product code identifying make and model of phone.
  - Phone_Infos is the information associated with a SKU
    - **SKU:** Primary key. Each product has a SKU to identify what it is. Not automatically generated.
    - **Manufacturer**: A varchar that holds the name of the manufacturer of the product.
    - **Model**: A varchar that holds the model# of the product. This is usually how the manufacturer uniquely identifies their product. Required and non-null.
    - **Phone_Name**: A varchar that holds a string describing the model of the phone in detail. Not required. Can be null.
    - **Phone_Price**: An integer that holds the price of the phone. Can be null. Not required. If null, that means it is discontinued.
  - Service plans are the other things sold in stores:
    - **Plan_ID**: Primary key, auto-incrementing integer. This will identify each phone's service plan (each line).
    - **Plan_Cost**: The cost of the given service plan in USD. stored as an integer and is required to be non-null.
    - **Plan_Name**: The designated name of a plan, stored as a 255 byte varchar.
    - **Plan_Desc**: A brief description of the plan, stored as a 255 byte varchar.
- **Relationships**

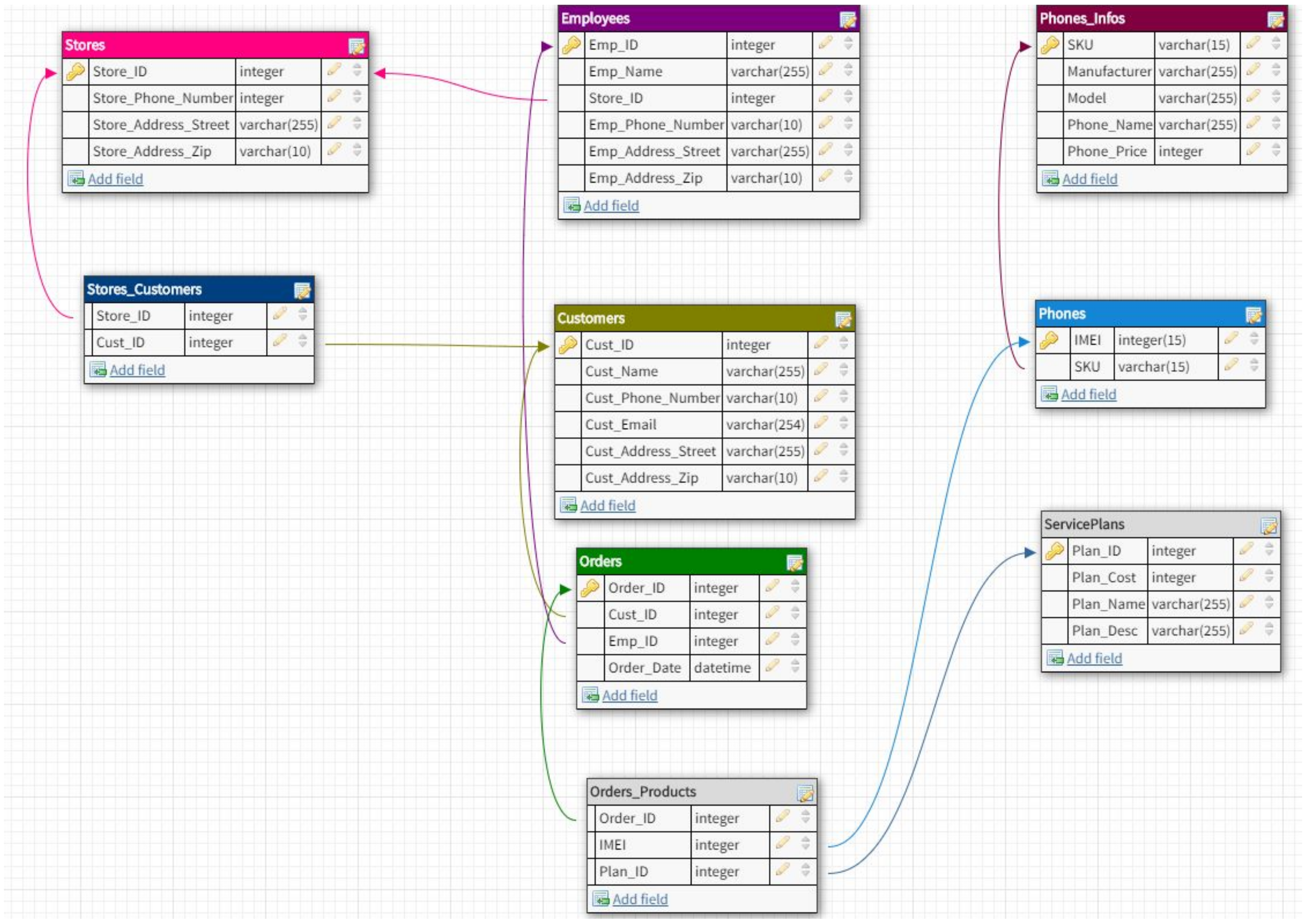
- Stores-Customers
  - A store may have 0 or many customers. When a store first opened they have 0 customers. As more people buy from the store, more customers will be associated with that store. A customer may belong to 1 or more stores (cannot be 0 because they would not be a customer at that point).
- Stores-Employees
  - A store may have 0 or many employees. When a store first is built, it may not have any employees associated with it yet. An employee can belong to 1 and only 1 store.
- Customers-Orders
  - A customer may have 1 or many orders. An order must belong to 1 and only 1 customer.
- Employees-Orders
  - An employee may have sold 0 to many orders. If employee is new, they will have sold 0 orders. An order must be sold by 1 and only 1 employee.
- ServicePlans-Orders
  - A service plan may be associated with 1 and only 1 order. An order may have 0 or more plans on it. (0 service plans if customer purchases an unlocked phone).
- Phones-Orders
  - A phone may be associated with 1 and only 1 order. An order can only be placed if a customer purchases at least 1 phone.
- Phones-ServicePlans
  - A phone may be associated with at most 1 service plan - an unlocked phone may be purchased without a plan. A service plan must be associated with a phone.

Entity-Relationship-Diagram:



Xiao Kuang
Logan Miller

Schema:



Stores		
🔑	Store_ID	integer
	Store_Phone_Number	integer
	Store_Address_Street	varchar(255)
	Store_Address_Zip	varchar(10)

Add field

Employees		
🔑	Emp_ID	integer
	Emp_Name	varchar(255)
	Store_ID	integer
	Emp_Phone_Number	varchar(10)
	Emp_Address_Street	varchar(255)
	Emp_Address_Zip	varchar(10)

Add field

Phones_Infos		
🔑	SKU	varchar(15)
	Manufacturer	varchar(255)
	Model	varchar(255)
	Phone_Name	varchar(255)
	Phone_Price	integer

Add field

Stores_Customers	
Store_ID	integer
Cust_ID	integer

Add field

Customers		
🔑	Cust_ID	integer
	Cust_Name	varchar(255)
	Cust_Phone_Number	varchar(10)
	Cust_Email	varchar(254)
	Cust_Address_Street	varchar(255)
	Cust_Address_Zip	varchar(10)

Add field

Phones		
🔑	IMEI	integer(15)
	SKU	varchar(15)

Add field

ServicePlans		
🔑	Plan_ID	integer
	Plan_Cost	integer
	Plan_Name	varchar(255)
	Plan_Desc	varchar(255)

Add field

Orders		
🔑	Order_ID	integer
	Cust_ID	integer
	Emp_ID	integer
	Order_Date	datetime

Add field

Orders_Products	
Order_ID	integer
IMEI	integer
Plan_ID	integer

Add field

Xiao Kuang
Logan Miller

b) Fixes based on Feedback from Previous Steps:

- Integer lengths were removed except for IMEI to ensure its 15 digit display length. Phone numbers, zips, and sku's were switched to varchars.
- Added linking tables to link together the orders and the phones & plans associated with an order.
- Renamed plan_infos table to serviceplans and got rid of the first version of serviceplans because all of those relationships are now held in the orders_products and orders table.
- Order_id was removed from the phones table because that is now represented by the order_products table.
- In addition to the smaller fixes like discrepancies in which fields were allowed to be NULL between our actual commands and our descriptions, there were two main things that have been added to the final draft. First, the explicit HTML input boxes have been defined for all of the desired features. Each feature is going to get it's own page that is linked to from the index and is rendered with an Express/Handlebars server. These features include things like creating a new order or looking up an existing customer. Ultimately all of the technical requirements for accessing and manipulating all of the tables will be accomplished by some combination of these features.
- The second big update was to the depth of the select commands. Rather than a static view of all of the entries in each table we now have commands that will enable more useful results from table lookups. These include things like looking up customers with dynamic filters and following foreign keys to find all order/phone information for a given customer. We believe these changes will result in a more fully functional and realistically useful interface.
- Other misc. Items changed was to update attributes in outline regarding whether or not they can be null or required.

Xiao Kuang
Logan Miller

One .SQL File should contain:

a) Data Definition Queries

```
-- phpMyAdmin SQL Dump
-- version 4.8.5
-- https://www.phpmyadmin.net/
--
-- Host: classmysql.engr.oregonstate.edu:3306
-- Generation Time: Feb 10, 2019 at 08:15 PM
-- Server version: 10.1.22-MariaDB
-- PHP Version: 7.0.33

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `cs340_kuangx`
--

-- --------------------------------------------------------

--
-- Table structure for table `Customers`
--

CREATE TABLE `Customers` (
  `Cust_ID` int(11) NOT NULL,
  `Cust_Name` varchar(255) NOT NULL,
  `Cust_Phone_Number` varchar(10) NOT NULL,
  `Cust_Email` varchar(254) NOT NULL,
  `Cust_Address_Street` varchar(255) DEFAULT NULL,
  `Cust_Address_Zip` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Customers`
--

INSERT INTO `Customers` (`Cust_ID`, `Cust_Name`, `Cust_Phone_Number`, `Cust_Email`,
`Cust_Address_Street`, `Cust_Address_Zip`) VALUES
(1, 'John Allen', '1038859123', 'AllenJ@oregonstate.edu', '1523 Mcleod Ave', '97330'),
(2, 'Richard Richardson', '5037678823', 'RR@gmail.com', '11542 NW Johnson Stâ€™', '92232'),
(3, 'Tom Allen', '5037748812', 'AllenT@oregonstate.edu', '1523 Mcleod Ave', '97330'),
(4, 'Taylor Brooks', '1203391842', 'brooksy@yahoo.com', '4273 Linkford Pk', '44723');

-- --------------------------------------------------------

--
-- Table structure for table `Employees`
--

CREATE TABLE `Employees` (
  `Emp_ID` int(11) NOT NULL,
  `Emp_Name` varchar(255) NOT NULL,
  `Store_ID` int(11) NOT NULL,
  `Emp_Phone_Number` varchar(10) NOT NULL,
  `Emp_Address_Street` varchar(255) NOT NULL,
  `Emp_Address_Zip` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Employees`
--

INSERT INTO `Employees` (`Emp_ID`, `Emp_Name`, `Store_ID`, `Emp_Phone_Number`,
`Emp_Address_Street`, `Emp_Address_Zip`) VALUES
(1, 'Chuck Smith', 1, '1413777283', '1112 Street St', '97330'),
(2, 'Larry Gurgitch', 2, '1541883912', '1455 Park Ave', '97330'),
(3, 'Mary Flatt', 2, '1142325542', '1185 Pickford St', '92333');

-- --------------------------------------------------------

--
-- Table structure for table `Orders`
--

CREATE TABLE `Orders` (
  `Order_ID` int(11) NOT NULL,
  `Cust_ID` int(11) NOT NULL,
  `Emp_ID` int(11) NOT NULL,
  `Order_Date` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Orders`
--

INSERT INTO `Orders` (`Order_ID`, `Cust_ID`, `Emp_ID`, `Order_Date`) VALUES
(1, 1, 1, '2018-04-23 12:04:24'),
(2, 2, 1, '2018-07-11 08:09:48'),
(3, 3, 3, '2018-12-27 15:10:28'),
(4, 4, 2, '2019-01-27 13:56:43');

-- --------------------------------------------------------
```

Xiao Kuang
Logan Miller

```
--
-- Table structure for table `Orders_Products`
--

CREATE TABLE `Orders_Products` (
  `Order_ID` int(11) NOT NULL,
  `IMEI` int(11) NOT NULL,
  `Plan_ID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Orders_Products`
--

INSERT INTO `Orders_Products` (`Order_ID`, `IMEI`, `Plan_ID`) VALUES
(1, 486723455, 4),
(2, 112999908, 1),
(3, 227823009, 1),
(4, 368899013, 4);

-- --------------------------------------------------------

--
-- Table structure for table `Phones`
--

CREATE TABLE `Phones` (
  `IMEI` int(15) NOT NULL,
  `SKU` varchar(15) NOT NULL,
  `Order_ID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Phones`
--

INSERT INTO `Phones` (`IMEI`, `SKU`, `Order_ID`) VALUES
(112999908, '6282526', 3),
(227823009, '6282526', 4),
(368899013, '6303049', 1),
(486723455, '6009657', 2);

-- --------------------------------------------------------

--
-- Table structure for table `Phones_Infos`
--

CREATE TABLE `Phones_Infos` (
  `SKU` varchar(15) NOT NULL,
  `Manufacturer` varchar(255) NOT NULL,
  `Model` varchar(255) NOT NULL,
  `Phone_Name` varchar(255) DEFAULT NULL,
  `Phone_Price` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Phones_Infos`
--

INSERT INTO `Phones_Infos` (`SKU`, `Manufacturer`, `Model`, `Phone_Name`, `Phone_Price`) VALUES
('6009657', 'Apple', 'MT5C2LL/A', 'iPhone XS Max 64GB - Gold ', 1100),
('6282526', 'Samsung', 'SM-N960U', 'Galaxy Note9 128GB - Ocean Blue', 1000),
('6303049', 'Google', 'GA00468-US', 'Pixel 3 128GB Not Pink', 929);

-- --------------------------------------------------------

--
-- Table structure for table `ServicePlans`
--

CREATE TABLE `ServicePlans` (
  `Plan_ID` int(11) NOT NULL,
  `Plan_Cost` int(11) NOT NULL,
  `Plan_Name` varchar(255) NOT NULL,
  `Plan_Desc` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `ServicePlans`
--

INSERT INTO `ServicePlans` (`Plan_ID`, `Plan_Cost`, `Plan_Name`, `Plan_Desc`) VALUES
(1, 0, 'No Plan', 'No Plan'),
(2, 20, 'No Data', 'Unlimited call and texting but no data service'),
(3, 20, 'Limited Data', 'Unlimited call and text with 2GB of data'),
(4, 60, 'Unlimited Data', 'Unlimited call, text, and data');

-- --------------------------------------------------------

--
-- Table structure for table `Stores`
--

CREATE TABLE `Stores` (
  `Store_ID` int(11) NOT NULL,
  `Store_Phone_Number` int(11) NOT NULL,
  `Store_Address_Street` varchar(255) NOT NULL,
  `Store_Address_Zip` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Stores`
--
```

Xiao Kuang
Logan Miller

```
INSERT INTO `Stores` (`Store_ID`, `Store_Phone_Number`, `Store_Address_Street`,
`Store_Address_Zip`) VALUES
(1, 1036782932, '566 Newburg Dr', '97330'),
(2, 1036788402, '1100 Stent Ln', '97330'),
(3, 1009582911, '521 Lovelace St', '92333');

-- --------------------------------------------------------

--
-- Table structure for table `Stores_Customers`
--

CREATE TABLE `Stores_Customers` (
  `Store_ID` int(11) NOT NULL,
  `Cust_ID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `Stores_Customers`
--

INSERT INTO `Stores_Customers` (`Store_ID`, `Cust_ID`) VALUES
(1, 1),
(3, 2),
(2, 3),
(1, 4);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `Customers`
--
ALTER TABLE `Customers`
  ADD PRIMARY KEY (`Cust_ID`),
  ADD UNIQUE KEY `Cust_ID` (`Cust_ID`);

--
-- Indexes for table `Employees`
--
ALTER TABLE `Employees`
  ADD PRIMARY KEY (`Emp_ID`),
  ADD UNIQUE KEY `Emp_ID` (`Emp_ID`),
  ADD KEY `Employees_fk0` (`Store_ID`);

--
-- Indexes for table `Orders`
--
ALTER TABLE `Orders`
  ADD PRIMARY KEY (`Order_ID`),
  ADD UNIQUE KEY `Order_ID` (`Order_ID`),
  ADD KEY `Orders_fk0` (`Cust_ID`),
```

Xiao Kuang
Logan Miller

```
  ADD KEY `Orders_fk1` (`Emp_ID`);

--
-- Indexes for table `Orders_Products`
--
ALTER TABLE `Orders_Products`
  ADD KEY `Orders_Products_fk0` (`Order_ID`),
  ADD KEY `Orders_Products_fk1` (`IMEI`),
  ADD KEY `Orders_Products_fk2` (`Plan_ID`);

--
-- Indexes for table `Phones`
--
ALTER TABLE `Phones`
  ADD PRIMARY KEY (`IMEI`),
  ADD UNIQUE KEY `IMEI` (`IMEI`),
  ADD KEY `Phones_fk0` (`SKU`),
  ADD KEY `Phones_fk1` (`Order_ID`);

--
-- Indexes for table `Phones_Infos`
--
ALTER TABLE `Phones_Infos`
  ADD PRIMARY KEY (`SKU`);

--
-- Indexes for table `ServicePlans`
--
ALTER TABLE `ServicePlans`
  ADD PRIMARY KEY (`Plan_ID`);

--
-- Indexes for table `Stores`
--
ALTER TABLE `Stores`
  ADD PRIMARY KEY (`Store_ID`),
  ADD UNIQUE KEY `Store_Phone_Number` (`Store_Phone_Number`);

--
-- Indexes for table `Stores_Customers`
--
ALTER TABLE `Stores_Customers`
  ADD KEY `Stores_Customers_fk0` (`Store_ID`),
  ADD KEY `Stores_Customers_fk1` (`Cust_ID`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `Customers`
--
ALTER TABLE `Customers`
```

Xiao Kuang
Logan Miller

```
  MODIFY `Cust_ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT for table `Employees`
--
ALTER TABLE `Employees`
  MODIFY `Emp_ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO_INCREMENT for table `Orders`
--
ALTER TABLE `Orders`
  MODIFY `Order_ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT for table `Phones`
--
ALTER TABLE `Phones`
  MODIFY `IMEI` int(15) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=486723456;

--
-- AUTO_INCREMENT for table `ServicePlans`
--
ALTER TABLE `ServicePlans`
  MODIFY `Plan_ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT for table `Stores`
--
ALTER TABLE `Stores`
  MODIFY `Store_ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `Employees`
--
ALTER TABLE `Employees`
  ADD CONSTRAINT `Employees_fk0` FOREIGN KEY (`Store_ID`) REFERENCES `Stores` (`Store_ID`);

--
-- Constraints for table `Orders`
--
ALTER TABLE `Orders`
  ADD CONSTRAINT `Orders_fk0` FOREIGN KEY (`Cust_ID`) REFERENCES `Customers` (`Cust_ID`),
  ADD CONSTRAINT `Orders_fk1` FOREIGN KEY (`Emp_ID`) REFERENCES `Employees` (`Emp_ID`);

--
-- Constraints for table `Orders_Products`
--
ALTER TABLE `Orders_Products`
```

Xiao Kuang
Logan Miller

```
  ADD CONSTRAINT `Orders_Products_fk0` FOREIGN KEY (`Order_ID`) REFERENCES `Orders`
(`Order_ID`),
  ADD CONSTRAINT `Orders_Products_fk1` FOREIGN KEY (`IMEI`) REFERENCES `Phones` (`IMEI`),
  ADD CONSTRAINT `Orders_Products_fk2` FOREIGN KEY (`Plan_ID`) REFERENCES `ServicePlans`
(`Plan_ID`);

--
-- Constraints for table `Phones`
--
ALTER TABLE `Phones`
  ADD CONSTRAINT `Phones_fk0` FOREIGN KEY (`SKU`) REFERENCES `Phones_Infos` (`SKU`),
  ADD CONSTRAINT `Phones_fk1` FOREIGN KEY (`Order_ID`) REFERENCES `Orders` (`Order_ID`);

--
-- Constraints for table `Stores_Customers`
--
ALTER TABLE `Stores_Customers`
  ADD CONSTRAINT `Stores_Customers_fk0` FOREIGN KEY (`Store_ID`) REFERENCES `Stores`
(`Store_ID`),
  ADD CONSTRAINT `Stores_Customers_fk1` FOREIGN KEY (`Cust_ID`) REFERENCES `Customers`
(`Cust_ID`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

URL to an index.html page

http://web.engr.oregonstate.edu/~millelog/

people.oregonstate.edu/~kuangx/CS340/index.html

```
<!DOCTYPE html>

<html>

<head>

        <title>Index for Cellular Company</title>

</head>

<body>

        <h1>Index for Cellular Company</h1>
        <div><a href="">Add new order</a></div>
        <div><a href="">Add new customer</a></div>
        <div><a href="">Look up customer</a></div>
        <div><a href="">Update customer</a></div>
        <div><a href="">Add phone to inventory</a></div>
        <div><a href="">Hire employee</a></div>
        <div><a href="">Fire employee</a></div>
        <div><a href="">Add new store</a></div>
        <div><a href="">Create new service plan</a></div>
</body>
</html>
```

One .SQL file should contain the Data Manipulation Queries:

These are the queries that your website uses to let your users interact with data. Thus SELECT, INSERT, UPDATE and DELETE queries to provide the functionalities described in the CS340 Project Guide

- Your database should be pre-populated with sample data.
- Your database should have at least 4 entities and at least 4 relationships, one of which must be a many-to-many relationship.
- It should be possible to INSERT entries into every table individually.

```
INSERT INTO stores (store_phone_number, store_address_street,
store_address_zip) VALUES (:store_phone_input,
:store_address_street_input, :store_address_zip_input);

INSERT INTO employees (emp_name, store_id, emp_phone_number,
emp_address_street, emp_address_zip) VALUES  (:emp_name_input,
:store_id_input, :emp_phone_number_input, :emp_address_street_input,
:emp_address_zip_input);

INSERT INTO customers (cust_name, cust_phone_number, cust_email,
cust_address_street, cust_address_zip) VALUES  (:cust_name_input,
:cust_phone_number_input, :cust_email_input, :cust_address_street_input,
:cust_address_zip_input);

INSERT INTO orders (cust_id, emp_id, order_date) VALUES (:cust_id_input,
:emp_id_input, :order_date_input);

INSERT INTO phone_infos (sku, manufacturer, model, phone_name,
phone_price) VALUES (:sku_input, :manufacturer_input, :model_input,
:phone_name_input, :phone_price_input);

INSERT INTO phones (imei, sku) VALUES (:imei_input, :sku_input);

INSERT INTO serviceplans (plan_cost, plan_name, plan_desc) VALUES
(:plan_cost_input, :plan_name_input, :plan_desc_input);

INSERT INTO stores_customers (store_id, cust_id) VALUES
(:stores_id_input, :cust_id_input);

INSERT INTO order_products (order_id, imei, plan_id) VALUES
(:order_id_input, :imei_input, :plan_id_input);
```

Xiao Kuang
Logan Miller

- Every table should be used in at least one SELECT query. For the SELECT queries, it is fine to just display the content of the tables, but your website needs to also have the ability to search using text or filter using a dynamically populated list of properties. This search/filter functionality should be present for at least one entity. It is generally not appropriate to have only a single query that joins all tables and displays them.

```
-- ////////////
-- what to select from stores?

-- 1. Select all info for all stores
            -- Used to list the stores in the manage page
SELECT * FROM Stores;

-- 2.

-- ////////////
-- what to select from customers?
-- 1. Customer lookup, would only need to look up their phone number and
verify the name matches
SELECT Cust_Name, Cust_Phone_Number FROM Customers
WHERE Cust_Name=:Cust_Name_Search
OR Cust_Phone_Number=:Cust_Phone_Search
OR Cust_Email=:Cust_Email_Search;

-- 2. Show customer name, their service plan information(s)
-- add WHERE clause to filter
SELECT c.Cust_Name, sp.Plan_Name, sp.Plan_Desc, sp.Plan_Cost FROM
Customers c
inner join Orders o on o.Cust_ID = c.Cust_ID
inner join Orders_Products op on op.Order_ID = o.Order_ID
inner join ServicePlans sp on op.Plan_ID = sp.Plan_ID;

-- 3. Show customer name, their phone information(s)
-- add WHERE clause to filter
SELECT c.Cust_Name, pi.Manufacturer, pi.Phone_Name FROM Customers
c
inner join Orders o on o.Cust_ID = c.Cust_ID
inner join Orders_Products op on op.Order_ID = o.Order_ID
inner join Phones p on op.IMEI = p.IMEI
inner join Phones_Infos pi on p.SKU = pi.SKU;


-- 4. Show customer name, when they made that order, and what they
purchased
-- add WHERE clause to filter
select c.Cust_Name, o.Order_ID, o.Order_Date, sp.Plan_Name,
pi.Phone_Name from Customers c
inner join Orders o on o.Cust_ID = c.Cust_ID
inner join Orders_Products op on op.Order_ID = o.Order_ID
inner join Phones p on op.IMEI = p.IMEI
inner join Phones_Infos pi on p.SKU = pi.SKU
inner join ServicePlans sp on op.Plan_ID = sp.Plan_ID;

-- /////////////
-- what to select from employees?

-- 1. Show which employees belong to which stores
-- add WHERE clause to filter
select e.Emp_Name, s.Store_ID from Stores s
inner join Employees e on e.Store_ID = s.Store_ID
order by e.Emp_Name asc;

-- 2. Show how many orders each employee has sold
-- add WHERE clause to filter
select e.Emp_Name, COUNT(*) as orderQty from Employees e
inner join Orders o on o.Emp_ID = e.Emp_ID
group by e.Emp_Name;

-- /////////////
-- what to select from phones and phones_infos

-- 1. List each phones sorted by sku
select pi.SKU, p.IMEI, pi.Manufacturer, pi.Phone_Name from Phones p
inner join Phones_Infos pi on p.SKU = pi.SKU;
```

Xiao Kuang
Logan Miller

```
-- 2. List Phone_Infos
select * from Phones_Infos;

-- misc
select * from ServicePlans;
select * from Phones;
select * from Phones_Infos;
select * from Employees;
select * from Stores;
select * from Orders;
select * from Orders_Products;
```

- You need to include one DELETE and one UPDATE function in your website, for any one of the entities. In addition, it should be possible to add and remove things from at least one many-to-many relationship and it should be possible to add things to all relationships.

```
UPDATE customers SET cust_name=:cust_name_update,
cust_phone_number=:cust_phone_number_update,
cust_email=:cust_email_update,
cust_address_street=:cust_address_street_update,
cust_address_zip=:cust_address_zip_update WHERE
cust_id=:cust_id_update;

DELETE FROM customers WHERE cust_id=:cust_id_delete;

UPDATE orders_products SET plan_id=:plan_id_update WHERE
order_id=:order_id_update;

DELETE FROM orders_products WHERE order_id=:order_id_delete AND
imei=:imei_delete
```

- In a many-to-one relationship (like bsg_people to bsg_planets), you should be able to set the homeworld value to NULL (such as on a person in bsg_people). That removes the relationship.

UPDATE employees SET store_id=NULL WHERE emp_id=:emp_id_update;

- In a many-to-many relationship, to remove a relationship one would need to delete a row from a table. That would be the case with bsg_people and bsg_certifications. One should be able to add and remove certifications for a person without deleting either bsg_people rows or bsg_certification rows.

  DELETE FROM stores_customers WHERE cust_id=:cust_id_delete_store AND store_id=:store_id_delete_cust;

  DELETE FROM orders_products WHERE plan_id=:plan_id_delete;